GWD-R (Recommendations Track)
Grid Protocol Architecture Working Group
http://grid.lbl.gov/GPA/

John Brooke and Donal Fellows
University of Manchester
Feb 17th 2003

Draft Discussion Document for GPA-WG

Abstraction of functions for resource brokers.

Working Draft, Version 1.0

Author(s)	Institution(s)
John Brooke Donal Fellows	University of Manchester, UK, j.m.brooke@man.ac.uk, donal.fellows@man.ac.uk

Abstract

This document is proposed as part of the discussion in the Grid Protocol Architecture Research Group on Core Functions for Production Grids. It tries to identify the functions required for effective resource brokerage on an architectually heterogeneous Grid. It discusses the relation of such a task to the problem of devising a sufficiently rich means of description for grid resources. It describes a particular implementation of the ideas presented, namely the resource broker developed in the EU EuroGrid and GRIP projects. Finally it raises issues of the application of a Grid Resource Ontology. The aim of this document is to contribute to the discussions of Grid Resource Description which are taking place across a number of GGF research and working groups. It is considered particularly relevant to the current discussion in GPA-WG on Uniform Access to Computing Resources and links to Section 3.3 of the GGF draft on Core Funtions for Production Grids (available from the GPA-WG website. It also has relationship to the work of the CIM-WG, the GRAAP-WG and the Semantic Grid RG.

Table of Contents

1 Introduction

The primary GGF group for the Resource Broker Abstraction will be the Grid Protocol Architecture Working Group [1]. This is because the essence of resource brokerage as described here is a resource check, that the set of resources requested can be met by a particular site or machine. This is an important aspect of a Core Grid Function described as Uniform Access to Grid Resources. By this we mean that the composer of a Grid job should not need to know the details of the site in terms of policy or eventually the site architecture [2]. There are other important aspects of brokerage for which a broker has placeholders, namely reservation of resources (relevant GGF group is GRAAP-WG [3]), semantic description of resource (GGF Group Semantic Grid-RG [4]) and modelling of Grid resources (e.g. CIM-WG [5] and possibly others). Finally there is the question of the relation to OGSA so we present some preliminary ideas in the conclusion.

To summarise the arguments to be developed below we consider that an abstraction of a Grid resource broker will implement the following functionality.

- Resource discovery, the ability to discover resources on the Grid that can meet the clients request for resources. Note that this may involve several sites implementing a complex workflow, in the simplest case it means finding resource to run a single job.
- Resource capability checking against the users request, this involves checking that all resources including software, site policy, users authorization are in place to ensure that consignment of the job or sub-job would result in successful running at the site.
- Implementation of a Quality of Service policy, or the return of sufficient information for the client to make a selection of offers based on advertised QoS for the users request. An example of a QoS policy is that the job be completed and returned by a certain date and will cost less than X euros. We consider that in practice QoS cannot be separated from cost but the cost may be notional. Clearly a site could provide very good QoS by arbitrarily stopping all jobs on the system and running the users request to completion but it would certainly wish to be recompensed for implementing such a drastic interruption to its normal mode of functioning.
- Client – provider introduction, the provision of sufficient information or functionality for the client and the chosen service provider to have the basis for negotiating a secure contract that secures the interests of both parties. The broker may be involved in some way in these negotiations, e.g. by providing a trusted third party but it is not essential to the core functionality of the broker.

The structure of this paper is as follows. In Section 2 we discuss the fundamental concepts of brokerage that we are attempting to implement in the EUROGRID/GRIP Resource Broker [17,18] and why we consider that these concepts are important for the Core Grid functions associated with Uniform Access to computing resources. Section 3 discusses the inclusion of a scheduling policy and QoS requirements in resource brokerage and how this can be described in an abstract framework that is not dependent on the properties of particular Resource Management Software product (examples of RMS products are LSF, PBS Pro, IBM LoadLeveler). This section also discusses the return of information necessary for the negotiation of a client-provider contract. Section 4 discusses possibilities for a resource description language and refers to experience in GRIP in reconciling the UNICORE AJO framework with MDS. Section 5 summarises and also discusses future directions for this work and the possible relationship with OGSA.

2 Resource Brokerage and Uniform Access to Resources

2.1 The Importance of Uniform Access

Most current Grid access to computing resources involves sending a script to the remote systems and then either exec-ing a shell and passing the script to the shell, or, more commonly for a purely computing resource, submitting the script to a batch queuing system.

To be truly described as uniform computing access there has to be a means for composing a common structure for a request for computing resources that is independent of different policies / configurations and naming conventions at individual sites or machines. A request for a computing resource also implicitly involves (for example) a request for staging areas for files, areas for storage of results, temporary workspace while the request is running. A problem is that there is no agreed convention for providing this holding environment for the computing request and unless the request for resources is conformant with the local site policy[1] it will fail. To take just one example, supposing a job is submitted under a temporary account validated by some certificate mapping policy. If the site has a policy that all filestore is deleted at the end of the running of the job, then a request that assumes that the files are held to be post-processed at a later stage will lose all such results. This also applies to an implicit assumption that pre-request staging is available. This becomes particularly important when we are chaining jobs or running workflow requests for computation and data access across several resource centers with dependencies between the different stages of the workflow. Clearly without pre and post compute resource file-staging such complex workflows cannot be guaranteed over multiple sites since we cannot usually guarantee that the next link in the job chain will be available before the last link ends its computation so that the results can be transferred to the next stage.

A request for a resource on the Grid therefore needs a working environment that is sufficiently robust to allow multi-site workflows to be constructed. These working environments need to be described in sufficiently abstract terms so that they can apply across a sufficiently wide range of policies. In other words the application execution environment for a Grid job needs to be described so that it can be implemented in multiple ways, on particular resources, implementing particular policies.

Even on a single resource, it must be possible to adapt to local policies (that may have nothing to do with the runtime / hosting environments) for jobs to run. It is important to note that not all workflow requests on the Grid are submitted to computers. A Grid workflow may include experimental or observational apparatus, e.g. telescopes.

However the key feature of a Uniform Computing Access service must be the ability to describe all features of the resource request (or job) including the essential features of the required execution environment in a manner which is independent of policies at individual sites. If this is not met then the individual policy of the site at which the component of the job request will be run must be determined as the job is being composed, this is not scalable and not resilient. In particular it makes migration of a job component to another site problematic since the policy differences may cause the component to fail unless renegotiation and alteration of the job scripts is carried out before migration. This would seem highly impractical. These problems are all compounded for multi-component workflows with dependencies between different components. The way the site policy impacts on the job component may well be dependent on the identity of the job owner. Authentication is via

[1] The term “policy” is used here in a general sense that includes site configuration, administrative based usage requirements, user authorization, etc. All of these must be taken into account in order to successfully run a job.

certificates but from this there must be access to information on the authorization rights of such a user at a given site. This, however, fits with the approach described in the example above since the Incarnation Data Base can hold such information, or pointers to where such information resides at the site. This concept has also been expressed as a "Policy Engine" [7] [8].

The essential point is that it is not sufficient for a Grid job request to state how many processing hours, memory etc.. is required, it is also essential that the request can be checked against the site policy and the user's authorization rights. In a truly scalable and dynamic GRID the user submitting the job will not wish to check this for all possible sites. This indicates the necessity of a Core Function, which we call *capability*, namely the checking that a request or job could potentially run at a given site or individual resource. We consider that checking for capability is a Core Function that should be implemented by a well-constructed resource broker. The alternatives are either that a user has an unnecessarily restricted view of a Grid mainly confined to sites where previous experience has shown that his/her jobs can run, or else that an unacceptable proportion of requests fail since the architectural, software and policy requests have not been checked before consignment of the job. Here are examples of these three separate types of failure,

- Architectural failure, a job request insufficient nodes for its memory needs since the memory per node is considerably less than the user anticipated. This might happen if a job that is normally run on a shared-memory node architecture (IBM pxxx, NEC SX-6) were re-routed to a machine where a node equated to a single processor with a small local memory (e.g. Cray T3E).
- Software failure, job requiring software resource X (e.g. Gaussian, Abaqus) is routed to a machine that although architecturally compatible with the job does not have the software resource installed or available to the particular user.
- Policy failure, jobs are chained so that the results of part 1 of the job run at Site A need to be held at the site until part 2 of the job run at Site B is ready to receiver them. The user expects the staged results to be stored for at least 24 hours in /tmp but site A has a policy (for perfectly good reasons) of deleting all space in /tmp relating to a job as soon as the job finishes.

Note that capability does not indicate that a job will run. A resource broker could report back that the clients job could be run at site X if the client registered and could give details of a URL for registration. Or the job could be run if payment of X dollars or Grid tokens (on a Grid where tokens for resource exchange were defined) was made and information of how to do this was returned to the client also. In this case the registration or payment would convert the potential into the actual and the job could be consigned with confidence. The EUROGRID resource broker [6] currently implements the first of these mechanisms (capability check and advertisement of registration information).

2.2 Resource Requestor and Resource Provider Spaces

We consider that that there exist two important groupings used within the concept of a Virtual Organization [9]. One is the group of Resource Requestors (RR) and the other the group of Resource Providers (RP). In the metacomputing usage of the Grid a Resource Requestor would normally be a users "job", i.e. a request for various processing, storage and transfer operations associated with some well defined task usually initiated from a single point (e.g. a specific workstation). The Resources Providers are the various servers, storage arrays, networks, that will perform the necessary data transformations implied by the job request. This is a one-to-many mapping from the space of RR to the space of RP since each

resource has its own RP space. In more sophisticated scenarios the RP space can recursively cast itself as an object in the RR space by passing on the resource request onwards as if it had become a Resource Requestor. See Figure 1 for an illustration of this process. The Globus and UNICORE Grid middleware systems with whose interoperability we are particularly concerned have a particular orientation to this mode of Grid usage. We note that the human usage of the Grid comes in here as an “invisible” factor. The one-to-many concept comes from a single person submitting a complex request. In the case of collaborative working, however, there is a many-to-many model and the interactions of the RR and RP spaces are highly dynamic. The Grid abstraction is particularly useful for examining such complex usage patterns since it allows each physical resource in the Grid to be used in either an RR or RP context. It is in this sense that one can genuinely consider the Access Grid [10] as a Grid in which each AG node is simultaneously and persistently in both the RR and RP spaces (full many-to-many interaction). The RR and RP spaces come naturally from the consideration of the implications of resource sharing in a Virtual Organization. We use the work “spaces” loosely, whether these can be considered as mathematically well-defined spaces (e.g. vector spaces) needs to be investigated.

We are principally concerned with the usage of Grids for solving complex computational tasks that may involve several computers. Along with the requests for computation comes requests for memory, disk space, networking bandwidth etc.. The specific problem we address is the conversion of a request in RR space into a request in RP space. We note that these requests may be composed by different middleware. An example is a user interacting with a portal for some application specific task, with the portal composing work requests sent to specific servers accessed via middleware such as Globus. The resource requestor implicitly asks for functions which will be implemented on the Grid. For example job submission is a function, writing to permanent storage is a function, file transfer is a function. These functions are defined by the nature of the users request, they may be accessed via Grid protocols calling on Grid services.

The separation of RR and RP spaces is important for Resource Brokers in that they allow the client to compose a request for a Grid job or workflow in a manner that is independent of detailed architectural or site policy considerations. The broker can then negotiate with resource providers to meet this request in different ways before it is translated into a request for a resource set described in the particular RP space of that site or machine. As a concrete example suppose a client composes a job using some application software, say a matrix manipulation package. The client is used to describing the scale of the problem in terms of the mathematical entities with which the package deals, e.g. the dimensions of the matrix, criteria for convergence such as required accuracy or maximum number of iterations of a numerical method. A specialist broker for such mathematical packages can take this request from RR space and map it onto a number of resource providers which it is able to discover. It will have special algorithms for translating from the mathematical RR space (maybe learning algorithms based on previous similar runs, maybe tables relating the performance of the underlying numerical algorithms on different processors and architectures). On a more mundane but nonetheless useful level, the broker may know about performance of a client job on architecture A and the client composes in an RR space for this architecture, so many nodes of architecture A, so much disk space, so much bandwidth between processors. The broker can translate this to Architecture B using general or learned information about comparative performance of A and B.

As an example of the interplay between RR and RP space see Figure 1. A request for resource consumption is formulated in RR space, this is forwarded to the RP spaces at B and C for realisation. At C a request needs to be referred onto D so there is composition of a request in RR space at C which is forwarded to RP space at D. Finally B needs to send a message to D to say that it has finished so that at D the whole workflow terminates and the initiator at A can be informed that the workflow has completed. For a concrete illustration suppose that the RR request at A comes from a user running MATLAB and that two numeric-

intensive tasks are farmed out to parallel resources at B and C with C needing to forward its output to a specialist visualization computer at D to compute and render an isosurface. Once D receives the message from B that it has finished and once it has completed its own part of the workflow it sends back bitmap files showing the isosurface to the MATLAB user at A where they are viewed, to the end user at A as merely part of his/her local MATLAB work.

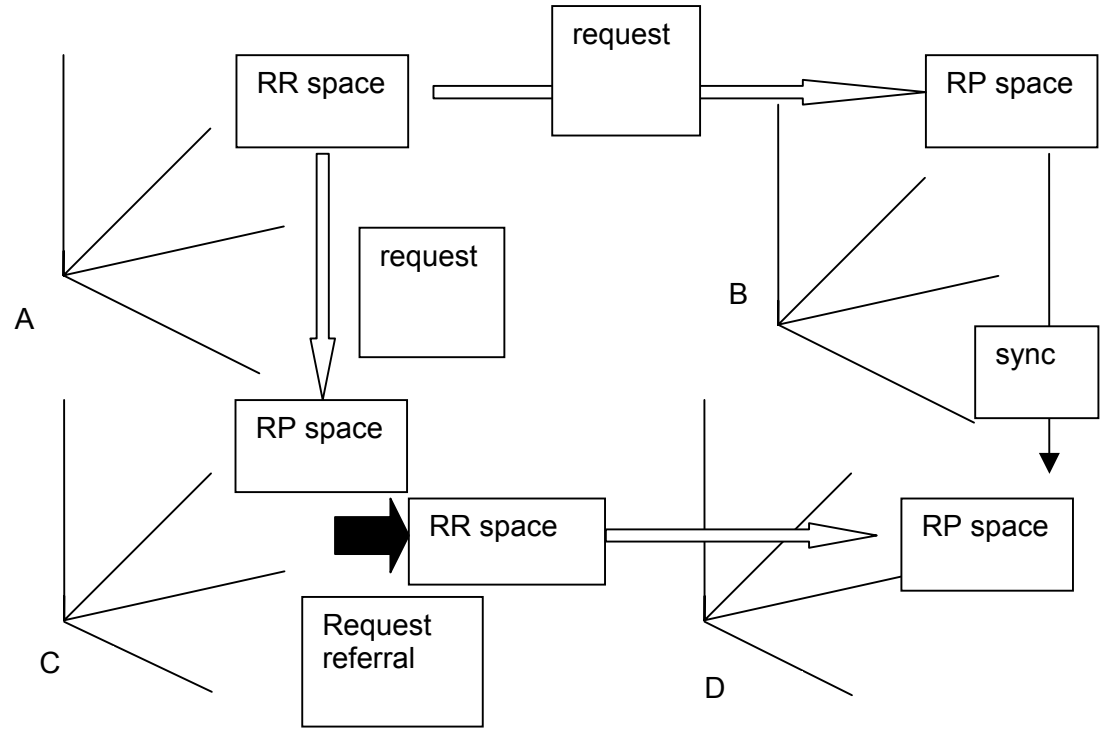

Figure 1: Request from RR space at A mapped into resource providers at B and C, with C forwarding a request formulated in RR space to RP space at D. B and C synchronize at end of workflow before results returned to the initiator A.

2.3 Summary of Capability Check Requirements

A resource broker needs to be able to check as many as possible of the following on behalf of a well-composed request by a client.

- Compatibility of the request with the architecture of the resource. We include in this the discovery of the resource.
- The job is framed so that the hosting environment of the resource can accommodate it.
- The client has the necessary authorization to run the task. If this is lacking the broker may indicate how it can be obtained, e.g. by registration or by payment.
- The provider has capacity to run the job, or is willing to make capacity available by negotiation.
- If the job is part of a workflow, that the staging of files or streaming of data can be accommodated between the co-operating resources on which the workflow will be instantiated.

This aspect of resource brokerage is called resource capability checking. If successful the broker can return to the client information needed to run the job, or can do this on the clients behalf once the client commits the job. We consider it important to allow for the forwarding of requests from one broker to another.

3 Quality of Service Requests

3.1 QoS and negotiation

Having identified a set of resource that are capable of running a client's work request the broker needs to ascertain whether a clients stated policy of quality of service can be accommodated and if so, what the cost will be. The cost could be purely nominal if the resources are being donated completely freely (this is unlikely in most circumstances), it could be notional as in the form of "resource tokens" to be drawn from a clients allowance within a Virtual Organization or it could be monetary. The QoS policy of the client could include,

- A time by which the work is to be completed and returned.
- A rate at which the work is to be done (this is implicit in parallel computing when one specifies the number of nodes or processors on which a job is to be run).
- A standard to which the work is to be performed, e.g. numerical accuracy, quality of image in visualization, frame rate in scientific animation or virtual reality application.
- Synchronicity requirements, e.g. that the computation be timed to be synchronous with another resource. An example of this would be processing information from a radio telescope [11] as soon as the telescope starts to receive signals from the chosen object processing should begin.

The broker may need to negotiate with the site or resource. The degree to which this can be done depends on factors outside the brokers control such as the scheduling mechanisms and degree of control of work which the sites RMS can accommodate. As an example we consider advanced scheduling. In the radio telescope example described above it may be known at what time observation will take place as time on such scarce and expensive resources is scheduled well in advance. The computational resource necessary for processing the signal needs to be co-scheduled. This will be particularly acute if the processing needs to be done in real time as for example is necessary in observing pulsars [11].

The GRAAP-WG of the GGF is investigating advanced reservation [3]. Clearly resource brokers wishing to perform this function on behalf of their clients will need to provide interfaces to such schedulers and cast the clients job request to include this if it is so desired. A uniform protocol for scheduling would make this task much easier.

From the point of view of the resource provider, a balance may need to be struck between the need to accommodate individual requests and the need to utilise resources efficiently in an overall sense. This involves the creation of Grid Economies that can include scheduling in an overall balance of resources with regard to time sharing, reservation, performance criteria. We consider that there will develop a symbiotic relationship between brokers and such Grid economies. Some work has already been done in this direction [12]. However we do not see this as the sole task of a broker, the factor of broker reliability provided by the resource capability checking described in Section 2 is also important. Therefore the broker may not always recommend or accept the "cheapest" offer. Brokers have an important role in

assuring the success of a Grid economy in that they have to balance between client and provider. Brokers that behave “badly”, e.g. do not return full information to clients resulting in jobs which fail, are eventually likely to be avoided or banned by both clients and resource providers.

3.2 Return of QoS information

We consider that QoS information be returned in a form that is separate from the resource capability check. In the EUROGRID broker which is used as the template for the GRIP interoperable resource broker, the resource check is done via a shadow AJO which is essentially of the same form as the actual job but returning a boolean value rather than full commital of the job. QoS offers are returned as tickets, these also have a stamp indicating the date and the period of validity. This latter is extremely important since QoS information is necessarily time-limited. When we consider the negotiation of the contract that the broker enables, the QoS ticket is an important element of this.

Tickets could be XML documents and standardization of protocols will be of importance in establishing QoS. These issues are discussed in Section 3.2 of [2]. The following functionality is recommended there as a Core Grid Service

- Establish a given, on-demand, virtual system relationship among an administratively independent set of Grid resources – that is, among all of the components that need to interoperate (e.g. computing resources for parallel, pipelined, multi-level processes) to accomplish a task in the Grid environment that involves many coordinated elements
- Return information sufficient for negotiation of a common QoS (e.g. time slot) among independent resources
- Returns information on cost of reservation.
- Returns information about possibility of pre-emption

The latter point, pre-emption is important. Also, the promised QoS may be difficult to maintain. We therefore propose the following as a desirable characteristic of a resource broker.

- Ability to monitor the QoS compliance of a client’s request, or to monitor and remember the behaviour of a site in terms of QoS compliance. In this way, either dynamically or else as information supplied before job submission, the client can judge how likely the QoS is going to be met and take appropriate action. In the dynamical, monitoring, case this may be by invoking a contract non-compliance condition of the contract with the provider and possibly migrating work elsewhere.

An example of a scheduling system which can provide some of the functionality outlined above is the Maui Silver Scheduler [13]. This operates as a meta-scheduler with interfaces to the PBS Resource Management Software (RMS). An example of a proposal for dynamical monitoring of contract compliance is given in [14]. For complex workflows involving multiple sites, scheduling is a very difficult problem and special intervention is often still necessary for critically important workflows, for example workflows that need to be scheduled around access to very expensive and costly experimental apparatus.

4 Resource Description Languages

4.1 Current RDLs

Current functioning Grid systems, e.g. Globus, UNICORE, Condor, all have had to develop some means of describing resources. However there is no common agreement as to what a Grid resource is. For example we can describe the architecture of a computer in terms of processor type and speed, number of processors, amount of memory, disk space etc.. However as indicated in Section 2, such information gives no guarantee that a users job will run on the computer. We also need information about the environment necessary for the job in terms of file areas, permissions, authorization, availability of software and software licences, bandwidth between components of a metacomputer. These issues are dealt with in Section 3.3 of [2] under the heading of Uniform Access to computing resources. We have described in Section 2 of this report how we envisage a resource broker would make a resource capability check as a first step in the brokerage service. We now examine the RDLs used by Globus and UNICORE to see how far they can supply a description rich enough for such a capability check.

Globus uses the Metadirectory Service (MDS) [15] to allow machine resources to be advertised. This allows for the resource discovery function of a broker to operate by querying the MDS and allows some discovery of architectural information. UNICORE contains in the Abstract Job Object as means of describing in abstract terms the resources that a job needs. This is translated into site specific information sufficient to create job submission scripts via the Incarnation Data Base (see Figure 2 for an example of Resource Description in the IDB). In Deliverable 2.4a of the GRIP project the ability to transfer the resource information in a UNICORE AJO to MDS is described. It is seen there that information about software and hosting environment resources cannot be translated to the MDS. The Globus GRAM provides for job submission and gives an interface to several RMS systems. A GRAM aware broker that provided interfaces for clients to compose jobs in terms of the GRAM resource specification language would thus provide a degree of abstraction necessary for GRID work. There appears to be no way to actually do the resource capability check described in Section 2, though the selection of a suitable site via querying the MDS and a well composed script for submission via GRAM would be likely to work in many cases.

Multi-site workflows are more problematic, since the environments on each host in the metacomputing network must provide facilities for pre and post staging of files and persistence to allow the network to function. The UNICORE AJO can capture such dependencies, it would be very useful to generalise this to other Grid systems. Condor [16] uses a match-making service which has many desirable features of a resource broker. In particular non-exact matching is permitted, thus the separation of Resource Requestor (RR) and Resource Provider (RP) space is implicit in Condor. We consider in the next section how this RR and RP separation can permit a novel approach to the resource description problem by the development of a Grid Resource ontology.

4.2 Semantic Grid approach

The considerations of Sections 2 and 3 of this report indicate that it would be a great advantage for resource brokers to reason about resources. For example a client job describes a resource set that cannot be exactly matched but could be run on a resource in a different way. An example would be a request that needed too much memory for any single node or processor on a system but could be run as a parallel job. Or, a given named software resource may not be present, but another differently named resource may provide functionality that is sufficiently equivalent for the clients request to be adequately run. In

terms of our formalism, an element of RR space may by mapped into multiple elements in a resources RP space, where all these RP elements can be considered equivalent in terms of satisfying the RR requirement. We now can see where a Grid Resource ontology might effectively operate, namely at the translation from RR to RP space.

More details of the Semantic Grid approach using ontologies can be found at [4]. For our purposes here, we can consider an ontology as a controlled vocabulary sufficiently rich to permit reasoning about the objects in the domain of the ontology. It thus is much richer than a dictionary, containing features appropriate to a thesaurus. It contains information about equivalences between the elements of the vocabulary. In our software resource example above the ontology would contain information to permit a resource broker to reason that the two differently named resources were equivalent.

It is very important not to consider ontology as a magical solution for the Grid Resource problem. The ontology would have to be developed and modified in the light of developing Grid experience. The evolution of the UNICORE Protocol Language (UPL) is a primitive version of what such an ontology might start to resemble. However the vertical structuring of UNICORE is restrictive in that only requests that can be formulated in terms of the AJO can be handled. For UNICORE to interoperate with other Grid systems such as Globus, more flexibility is needed. The GGF CIM-WG is investigating the use of the CIM language(Computer Interface Modelling) to provide a resource description framework. Since CIM was developed initially for a different purpose, namely describing the interfaces of components of a computer in such a way that they can interoperate, it is not clear that it will generalise to the soft resources such as site policy, hosting environments, software resources that have been shown to be essential for a resource capability check. However CIM needs to be carefully considered as it has attractive structural features for resource description.

We therefore propose to consider CIM in the light of providing a resource capability check and of modelling QoS information both of which we claim are crucial features of resource brokerage. In turn we intend to propose the UNICORE AJO as a test driver for CIM, we should be able to describe resources in the Incarnation Data Base in terms of CIM, if not then we have a problem in describing Grid workflows in such as way as to render them amenable to a capability check. Launching complex workflows that may fail will prove very serious for the functioning of a user-friendly Grid and will seriously impede the scalability of Virtual Organizations.

```
SOFTWARE_RESOURCE CPMDV3.0h
 SOFTWARE_RESOURCE CPMD      V3.4.1
 TEXTINFORESOURCE [ The      CPMD Pseudopotential Library ]
   TAG   [PP_LIBRARY_PATH ]
   VALUE [ /usr/local/cpmd/lib/PP_LIBRARY ]
 INVOCATION CPMD-V3.0h [
   PP_LIBRARY_PATH=/usr/local/cpmd/lib/PP_LIBRARY; \
        export PP_LIBRARY_PATH;
   /bin/mpprun -n $UC_PROCESSORS \ /usr/local/cpmd/bin/cpmd30h.x $CPMD_FILE
        $PP_LIBRARY
 ]
 INVOCATION CPMD-V3.4.1      [
  PP_LIBRARY_PATH=/usr/local/cpmd/lib/PP_LIBRARY; \
```

```
        export PP_LIBRARY_PATH;
     /bin/mpprun -n $UC_PROCESSORS \ /usr/local/cpmd/bin/cpmd3.4.1.x  $CPMD_FILE
          $PP_LIBRARY
    ]
```

Figure 2: Sample IDB definitions of a software resource, the Car-Parinello Molecular Dynamics (CPMD) software. Here we are able to provide two different versions of the software via the INVOCATION parameter.

5 Summary and Conclusions

To summarise, we consider that a resource broker should be able to

1. Discover resources and return to the client sufficient information to access them.
2. Provide a resource capability check, namely that the resource could run the clients workflow request. This may involve cooperation of multiple resources.
3. Negotiate a Quality of Service policy for a client and link this to cost. Return sufficient information for the negotiation of a contract to run the job according to an agreed QoS.
4. Monitor the compliance of the resource provider with the client request, either by monitoring dynamically or by learning about provider trustability.

In considering these points we were lead to the importance of a description framework or language for Grid resources and the advantage of a semantic approach via a Grid Resource ontology. A conceptualization of separate spaces for Resource Request description (RR space) and Resource Provider description (RP space) was found to be critical in locating the point in the broker architecture at which the ontology should be utilised. No one Grid system currently provides sufficient richness for all of the above functionality. However a comparison of the Globus and UNICORE resource description mechanisms can give very useful pointers to a future Resource Description framework or language. CIM is a possible candidate but it needs to be checked against some usage scenarios. We suggest that it should demonstrate sufficient richness to allow the incarnation of a UNICORE Abstract Job Object and to enable a capability check on the AJO via the methods described in the EUROGRID resource broker [6].

A protypical resource broker that performs a capability check on a simple UNICORE AJO via interrogation of a Globus MDS is now available [14]. This is describable in Java classes. A promising avenue of research is to translate the specification of the broker into XML. It is extensible in terms of providing placeholders for QoS information and thus can be a template for more sophisticated and application-specific brokers. It is also designed to allow broker to broker job transmission [6], this being important for a scalable Grid. This will be made available as open source and we will draw the attention of relevant GGF WGs and RGs to the broker description.

Future work is to develop a simple but working Grid Resource ontology and we will propose this to the Semantic Grid WG and CIM-WG. At a very minimum it should be able to duplicate the functionality of the Unicore Protocol Layer and to drive a UNICORE incarnation database (IDB).

6 Bibliography

1. Grid Protocol Architecture Working Group http://grid.lbl.gov/GPA
2. Johnston, W., Brooke, J., Core Grid Functions, GGF Document http://grid.lbl.gov/GPA
3. Grid Resource Allocation Agreement Protocol, GRAAP GGF http://www.gridforum.org/3_SRM/graap.htm
4. Semantic Grid GGF-RG http://www.semanticgrid.org
5. CIM-based Grid Schema GGF-WG http://www.gridforum.org/1_GIS/CIM.htm
6. EUROGRID – http://www.eurogrid.org, UNICORE http://www.unicore.de
7. Policy Engine: a framework for authorization, accounting policy specification and evaluation in Grids, B. Sundaram and B. M. Chapman. In GRID2001, 2nd IEEE Workshop on Grid Computing. 2001. Denver, Colorado, USA. http://www.cs.uh.edu/~ezgrid/PolicyEngine.pdf
8. EZ-Grid Project: resource brokerage for multi-site computing, EZ-Grid. http://www.cs.uh.edu/~ezgrid/arch.html
9. Foster, I, C. Kesselman, J. Nick, & S. Tuecke, 2001, “The Anatomy of the Grid”, International J. Supercomputer Applications, 15(3).
10. AccessGrid 2002, Access Grid Project – see collection of papers at www.accessgrid.org
11. Pickles, S.M., Brooke, J.M., Costen, F.C., Gabriel, E., Müller, M., Resch, M., Ord, S.M., 2001, Metacomputing across intercontinental networks, Future Generation Computer Systems 17, pp 911-18
12. Economy Grid, http://www.cs.mu.oz.au/~raj/grids/ecogrid/
13. Maui Silver Metascheduler. http://www.supercluster.org/documentation/silver/silveroverview.html
14. Vraalsen, F., Aydt, R.A., Medes, C.L., Reed, D.A., “Performance Contracts: Predicting and Monitoring Grid Application Behavior”, in GRID 2001, C.A.Lee (Ed.), LNCS 2242, Springer, pp 154-165
15. Czajkowski, S. Fitzgerald, I. Foster, & C. Kesselman, 2001, “Grid Information Services for Resource Sharing”, in HPDC-10, IEEE Press, information on MDS also available at http://www.golbus.org/mds
16. Condor – tools for high-throughput computing http://www.cs.wisc.edu/condor/

7 Glossary of Acronyms

- RR space: abstract space to represent requests for computational resource
- RP space: abstract space to represent provision of computational resource
- AJO: Abstract job object, abstraction of workflow for computation expressed as Java classes.
- Usite/Vsite In UNICORE an administrative domain is know as a Usite and a resource within the Usite is a Vsite. There may be many Vsites in a single Usite.
- NJS/IDB/TSI In UNICORE the Network Job Supervisor (NJS) translates the AJO into scripts by interacting with the Incarnation Data Base and these are then submitted to the Vsite via the Target System Interface (TSI)
- VO: Virtual Organization, set of resources shared to enable complex computations from a community of users.
- Globus, UNICORE: systems of Grid middleware for providing core Grid functions
- OGSA: Open Grid Services Architecture, set of interfaces and protocols proposed as Grid standard
- MDS and LDAP, Metacomputing Directory Service which uses Lightweight Directory Access Protocol to describe the resources available via Globus
- GRIS/GIIS Globus information servers GRIS (Grid Resource Information Service) at host level, GIIS (Grid Information Index Service) as aggregate directory.

8. Notices

8.1 Intellectual Property Statement

The GGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the GGF Secretariat.

The GGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the GGF Executive Director (see contacts information at GGF website).

8.2 Copyright Notice